CHAPTER

4

UNIX File Processing

case Your manager at Dominion Consulting, Rolfe Williams, wants you to extract a list of names from the phone number file you created earlier and sort the list. Then he wants you to help a Dominion client, Worldwide Hotels, design a new vendor and product report. The hotel firm wants to produce an alphabetical list of vendors and the products each offers. The vendor and product names reside in separate files: a product file and a vendor file. Both files contain vendor numbers. Rolfe asks you to use the UNIX file processing tools to produce a vendor report using the two files.

LESSON A

objectives

After studying this lesson, you should be able to:

- Explain UNIX's approach to file processing
- Use basic file manipulation commands
- Extract characters and fields from a file using the cut command
- Rearrange fields inside a record using the paste command
- Merge files using the sort command
- Create a new file by combining cut, paste, and sort

Extracting Information from Files

Now that you know how to work with UNIX files and editors, you're ready to learn how to manipulate files and work with their contents. After a brief discussion of UNIX file types and file structures, Lesson A defines file processing and shows you how to use redirection operators when processing files. You also learn how to manipulate files by creating, deleting, copying, and moving them to extract information from files, to combine fields, and to sort a file's contents, all in the context of the opening case. In Lesson B, you learn how to assemble information you extracted from files. You also create a script to automate a series of commands, link files with a common field, and use the awk command to format output. You complete these tasks to meet the goals of the opening case.

UNIX's Approach to File Processing

UNIX file processing is based on the idea that files should be treated as nothing more than character sequences. This concept of a file as a series of characters offers a lot of flexibility. Because you can directly access each character, you can perform a range of editing tasks, such as correcting spelling errors and organizing information as necessary.

Understanding UNIX File Types

Operating systems support several types of files. UNIX, like MS-DOS, has regular files, directories, and special files. **Regular files** contain information you create and manipulate, and include either ASCII files, such as the text files you created in Chapter 3, or binary files, such as those you create while compiling source code. You use regular files, also called **ordinary files**, in this chapter. Other file types include directories and special files, such as character files and block files. Chapter 2 explained that directories are system files for maintaining the structure of the file system. **Character special files** are related to serial input/output devices, such as printers. **Block special files** are related to devices, such as disks.

UNIX File Structures

Files can be structured in several ways. For example, UNIX stores data, such as letters, product records, or vendor reports, in **flat ASCII files**. UNIX structures a file depending on the kind of data it stores, and recognizes three kinds of regular files: unstructured ASCII characters, records, and trees. Figure 4-1 illustrates the three kinds of regular files.

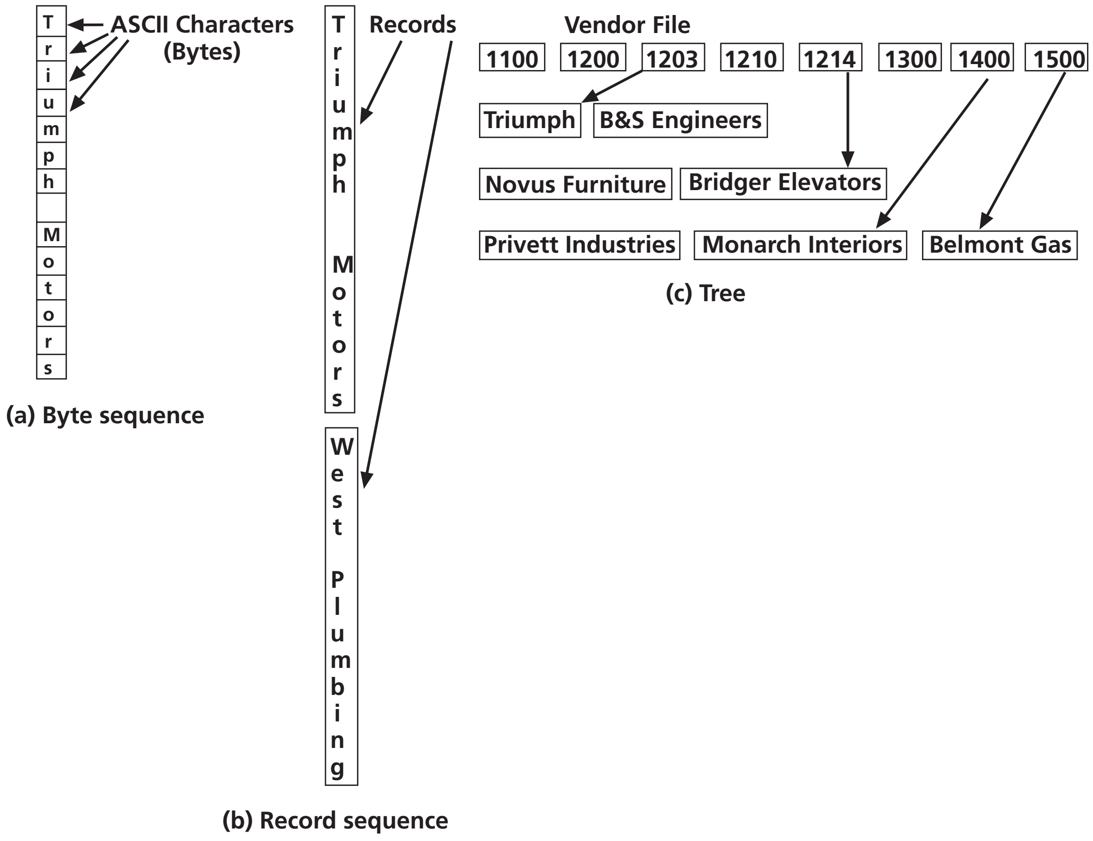

Figure 4-1: Three kinds of regular files

Figure 4-1(a) shows a file that is an unstructured sequence of bytes and is a typical example of a text file. This file structure gives you the most flexibility in data entry, because you can store any kind of data in any order. However, you can only retrieve the data in the same order, which may limit its overall usefulness. For example, suppose you list Worldwide Hotel's vendors in an unstructured ASCII file. You can only view or print the entire list, not just the vendor names or vendor numbers.

Figure 4-1(b) shows data as a sequence of fixed-length records, each having some internal structure. In this structure, UNIX reads the data as fixed-length records. Although you must enter data as records, you can also manipulate and retrieve the data as records. For example, you can select only certain vendor records to retrieve from the file.

The third kind of file, illustrated in Figure 4-1(c), is structured as a tree of records, not necessarily of the same length. Each record contains a key field, such as a record number, in a fixed position in the record. The key field sorts the tree, so you can quickly search for a record with a particular key. For example, you can quickly find the record for Triumph Motors by searching for record #1203.

Processing Files

When performing commands, UNIX **processes** data—it receives input from the standard input device (your keyboard) and then sends output to the standard output device (your monitor). System administrators and programmers refer to standard input as **stdin**. They refer to standard output as **stdout**. (The third standard output is called **standard error**, or **stderr**. When UNIX detects errors in processing system tasks and user programs, it directs the errors to stderr, which is the screen by default.)

You can use the redirection operators to save the output of a command or program in a file, or use a file as an input to a process. The redirection operators, therefore, help you process files.

Using Input and Error Redirection

You can use redirection operators (>, >>, 2>, <, and <<) to retrieve input from something other than the standard input device and to send output to something other than the standard output device.

You already used the output redirection operators in Chapter 1 when you created a new file by redirecting the output of several commands to files. Redirect output when you want to store the output of a command or program in a file. For example, recall that you can use the ls command to list the files in a directory, such as /home. The standard output device for the ls command is the screen, so you see the list on the screen. To redirect the list to a file called homedir.list, use the redirection symbol by typing ls > homedir.list.

You may also redirect the input to a program or command with the < operator. For instance, a program that accepts keyboard input may be redirected to read information from a file instead. In the following steps you create a file for the vi editor to read its commands from, instead of reading them from the keyboard.

To create a file from which the vi editor reads commands:

1 Use the vi editor, or the cat command with output redirection, to create the file testfile, containing the text:

```
This is line 1.
This is line 2.
This is line 3.
This is line 4.
```

2 In the vi editor,
Create another text file containing some vi commands as follows. Type **cat > commands** and press **Enter**. Type **dd** and press Enter. Type **H** and press **Enter**. Type **p** and press **Enter**. Type **:wq** and press **Enter**. Press **Ctrl+D**.

3 Type **cat commands** and press **Enter**. You see the contents of the commands file, which are:

```
dd
G
p
:wq
```

4 Type **vi testfile < commands** and press **Enter**. This loads testfile into the vi editor and redirects vi's input to the text in the command file. The text in the command file is treated as commands typed on the keyboard.

5 Type **cat testfile** and press **Enter**. You see the contents of testfile after the vi commands execute. The contents are:

```
This is line 2.
This is line 3.
This is line 4.
This is line 1.
```

You may also use the 2> operator to redirect commands or program error messages from the screen to a file.

To redirect error messages:

1 Force the ls command to display an error message by giving it an invalid argument. Assuming you have no file or directory in your /home directory named jojo, type **ls jojo** and press **Enter**. You see the error message:

```
ls: jojo: no such file or directory
```

2 Redirect the error output of the ls command. Type **ls jojo 2> errfile** and press **Enter**. There is no output on the screen.

3 Type **cat errfile** and press **Enter**. You see errfile's contents:

```
ls: jojo: no such file or directory
```

Manipulating Files

When you manipulate files, you work with the files themselves as well as their contents. This section explains how to complete the following tasks:

- Create files
- Delete files
- Copy files
- Move files
- Find files
- Display files
- Combine files
- Cut and paste file contents
- Sort files

Creating Files

The simplest way to create a file is to use the output redirection operator (>). You learned how to do this to redirect the cat command's output in Chapters 1–3. You can also use the redirection operator without a command to create an empty file.

To create an empty file:

1 After the $ command prompt, type **> newfile1** and press **Enter**.

This creates an empty file called newfile1.

2 To list the new file, type **ls –l newfile1** and press **Enter**.

You see only the information listed next, where jean is your user name.

```
-rw-rw-r--  1 jean  jean       0 Nov 1 16:57 newfile1
```

3 To create another new file, type **> newfile2** and press **Enter**.

You may also use the touch command to create empty files. For example, the following command creates the file newfile3, if the file does not already exist.

```
touch newfile3
```

The primary purpose of the touch command is to change a file's time and date stamp. UNIX maintains the following date and time information for every file:

- Creation date and time: the date and time the file was created
- Access date and time: the date and time the file was last accessed
- Modification date and time: the date and time the file was last modified

Although the touch command cannot change an existing file's creation date and time, it can alter the file's access and modification dates and times. By default, it uses the system date and time for the new values.

To create a file and alter its date/time stamp with the touch command:

1. Type **touch newfile3** and press **Enter**. This command creates the file newfile3.
2. Type **ls –l newfile3** and press **Enter**. You see a long listing for the newfile3 file. Note its modification date and time.
3. Wait at least one minute.
4. Type **touch newfile3** and press **Enter**. This updates the file's access and modification date and time stamps with the system date and time.
5. Type **ls –l newfile3** and press **Enter**. Look at the file's modification time. It should be different now.

Table 4-1 lists and describes the touch command's options.

Command option	Description
-a	Updates the access time only
-m	Updates the modification time only
-c	Prevents touch from creating the file if it does not exist

Table 4-1: Touch command options

Deleting Files

When you no longer need a file, you can delete it using the rm (remove) command. If you use rm without options, UNIX deletes the specified file without warning. Use the –i (interactive) option to have UNIX warn you before deleting the file. You can use the rm command to delete the new files you just created.

To delete a file from the current directory:

1. After the $ command prompt, type **rm newfile1** and press **Enter**.

   This permanently deletes newfile1 from the current directory.
2. Type **rm –i newfile2** and press **Enter**.

   You see the message, "remove 'newfile2'?".
3. Type **y** for yes and press **Enter**.

To delete a group of files using wildcards:

1. You can specify multiple filenames as arguments to the touch command. Type **touch file1 file2 file3 filegood filebad** and press **Enter**. This command creates the files file1, file2, file3, filegood, and filebad.
2. Type **ls file*** and press **Enter**. You see the listing for the files you created in Step 1.
3. Type **rm file*** and press **Enter**.
4. Type **ls** and press **Enter**. The files have been erased.

Removing Directories

When you no longer need a directory, you can use the rmdir command to remove it. The command takes the form:

Syntax rmdir *directory-name*

Note: A directory must be empty before you can remove it.

To remove a directory with the rmdir command:

1. Type **mkdir newdir** and press **Enter**. This creates a new directory named newdir.
2. Use a relative path with the touch command to create a new file in the newdir directory. Type **touch newdir/newfile** and press **Enter**. This creates the file newfile in the newdir directory.
3. Type **ls newdir** to see a listing of the newfile file.
4. To attempt to remove the directory, type **rmdir newdir** and press **Enter**. You see an error message similar to:

   ```
   rmdir: newdir: Directory not empty
   ```

5. Use a relative path with the rm command to delete newfile. Type **rm newdir/newfile** and press **Enter**.
6. The directory is now empty. Type **rmdir newdir** and press **Enter**.
7. Type **ls** and press **Enter**. The newdir directory is no longer there.

Copying Files

Chapter 2 introduced the cp command. Its general form is:

Syntax cp [-options] *source destination*

The command copies the file or files specified by the source path to the location specified by the destination path. You can copy files into another directory, with the copies keeping the same names as the originals. You can also copy files into another directory with the copies taking new names, or copy files into the same directory as the originals, with the copies taking new names.

For example, assume Tom is in his /home directory (/home/tom). In this directory he has the file reminder. Under his home directory he has another directory, duplicates (/home/tom/duplicates). He copies the reminder file to the duplicates directory with the following command:

```
cp reminder duplicates
```

After he executes the command, a file named reminder is in the duplicates directory. It is a duplicate of the reminder file in the /home/tom directory. Tom also has the file class_of_78 in his home directory. He copies it to a file named classmates in the duplicates directory with the following command:

```
cp class_of_78 duplicates/classmates
```

After he executes the command, the file classmates is stored in the duplicates directory. Although it has a different name, it is a copy of the class_of_78 file. Tom also has a file named memo_to_boss in his home directory. He wants to make a copy of it and keep the copy in his home directory. He types the following command:

```
cp memo_to_boss memo.safe
```

After he executes this command, the file memo.safe is stored in Tom's home directory. It is a copy of his memo_to_boss file.

You may specify multiple source files as arguments to the cp command. For example, Tom wants to copy the files project1, project2, and project3 to his duplicates directory. He types the following command:

```
cp project1 project2 project3 duplicates
```

After he executes the command, copies of the three files are stored in the duplicates directory. You may also use wildcard characters with the cp command. For example, Tom has a directory named designs under his home directory (/home/tom/designs). He wants to copy all files in the designs directory to the duplicates directory. He types the following command:

```
cp designs/* duplicates
```

After he executes this command, the duplicates directory contains a copy of every file in the designs directory.

The cp command is especially useful for preventing data loss: you can use it to make back-up copies of your files. You can create three new files and then copy them to a different directory. Then you can duplicate one file and give it a different name. Start by creating the subdirectory source in your home directory. You can create three new files and then copy them to the source directory.

To create three files and copy them to a directory:

1. If you do not already have a subdirectory source, make sure you're in your home directory and then create the directory. After the $ command prompt, type **mkdir source** and then press **Enter**.
2. To create three files in your home directory, type **> file1** and press **Enter**, type **> file2** and press **Enter**, and then type **> file3** and press **Enter**.
3. Now you can copy the three files to the source directory. Type **cp file1 file2 file3 source** and press **Enter**.

   Now you can copy of one of the files and give it a different name so you can distinguish it as a back-up file.
4. After the $ command prompt, type **cp file1 file1.sav** and press **Enter**.

   Now your working directory contains two files with identical contents but different names.

Recursively Removing Directories

The rm command normally requires that a directory be empty before you can remove it. This can be inconvenient when you need to remove a directory that has several subdirectories under it. The –r option, however, tells the rm command to remove a directory and everything it contains, including subdirectories. It even removes subdirectories of subdirectories. This operation is known as recursive removal.

To recursively remove a directory with several subdirectories:

1. Create a directory with several subdirectories. Type **mkdir company** and press **Enter**. Type **mkdir company/sales** and press **Enter**. Type **mkdir company/marketing** and press **Enter**. Type **mkdir company/accounting** and press **Enter**.
2. Create three empty files in the company directory. Type **touch company/file1 company/file2 company/file3** and press **Enter**.
3. Copy the files to the other directories. Type **cp company/file1 company/file2 company/file3 company/sales** and press **Enter**. Type **cp company/file1 company/file2 company/file3 company/marketing** and press **Enter**. Type **cp company/file1 company/file2 company/file3 company/accounting** and press **Enter**. (HINT: The three commands you just typed are very similar. You can reduce your typing by using the Up arrow key to recall the first command.)
4. Use the ls command to verify that the files were copied into all three directories.

5. Remove the company directory and everything it contains. **Type rm –r company** and press **Enter**.
6. Type **ls** and press **Enter**. The company directory is removed.

Caution: Be very careful with the rm –r command. It can permanently delete massive amounts of information.

Moving Files

Moving files is similar to copying them, except you remove them from one directory and store them in another. To move a file, use the mv (move) command. The command has the general form:

Syntax mv [-options] *source destination*

You can also use the mv command to rename a file by moving one file into another file with a different name.

To move a file from one directory to another:

1. To create the new file thisfile in your home directory, type **> thisfile** and then press **Enter**.
2. Type **mv thisfile source** and press **Enter** to move the new file to the source directory.
3. Type **ls** and press **Enter**. Thisfile is not listed. Type **ls source** and press **Enter**. You see thisfile listed.
4. To move more than one file, type the filenames before the directory name. For example, type **mv file1 file1.sav source** and press **Enter**.
5. To create the new file my_file, type **> my_file** and press **Enter**.
6. To rename my_file to your_file, type **mv my_file your_file** and press **Enter**.
7. Type **ls** and press **Enter**. You see your_file listed, but my_file is not listed.

Note: Moving and renaming a file are essentially the same operation.

You can also use the –i option with the mv command. It causes the command to prompt you before it overwrites an existing destination file.

Finding Files

The find command searches for files that have a specified name. Use the find command to locate files that have the same name or to find a file in any directory. The command has the form:

Syntax `find pathname –name filename`

Dissection

- ***pathname*** is the path name of the directory you want to search. The find command searches recursively; that is, it starts in the named directory and searches down through all files and subdirectories under the directory specified by *pathname.*
- **-name** indicates that you are searching for files with a specific filename. You may use wildcard characters in the filename. For example, you may use phone* to search for all filenames that begin with "phone." For other search conditions you can use with find, refer to Appendix B, "Syntax Guide to UNIX Commands."

The find command prohibits you from searching areas where you do not have system-level permissions. As the search progresses, it passes through protected areas, but you receive a "Permission denied" message each time you enter a directory for which you do not have access permissions.

You can use the find command to find every file named phone1 in the /home directory and all its subdirectories.

To find a file:

- After the $ command prompt, type **find /home –name phone1** and press **Enter**.

Note: Although Linux does not use it, other UNIX versions require the –print option after the filename to display the names of files the find command locates.

Combining Files

Now you're ready to work on the vendor report for Worldwide Hotels, Dominion's client. Two separate files, illustrated in Figure 4-2, store the data you need—product descriptions and vendor numbers.

```
File Name: products1

Lobby Furniture          1201
Ballroom Specialties     1221
Poolside Carts           1320
Formal Dining Specials   1340
Reservation Logs         1410
```

```
File Name: products2

Plumbing Supplies        1341
Office Equipment         1361
Carpeting Services       1395
Auto Maintenance         1544
Pianos and Violins       1416
```

Figure 4-2: Two product description files

You can use the cat command to combine the two files in a new vendor products master file. Do so by redirecting the output of cat to create the product1 file in your home directory. The file contains two colon-separated fields.

To use the cat command to combine files:

1 After the $ command prompt, type **cat > product1**.

2 Type the following text, pressing **Enter** at the end of each line:

Lobby Furniture:1201
Ballroom Specialties:1221
Poolside Carts:1320
Formal Dining Specials:1340
Reservation Logs:1410

3 Press **Ctrl+Z**.

Now you can redirect the output of cat to create the product2 file in your home directory. This file also contains two colon-separated fields.

4 After the $ command prompt, type **cat > product2**.

5 Type the following text, pressing **Enter** at the end of each line:

Plumbing Supplies:1423
Office Equipment:1361
Carpeting Services:1395
Auto Maintenance:1544
Pianos and Violins:1416

6 Press **Ctrl+Z**.

7 Now you can combine the two files in a master products file. After the $ command prompt, type **cat product1 product2 > products** and press **Enter**.

8 To list the contents of products, type **more products** and press **Enter**. You see the list:

```
Lobby Furniture:1201
Ballroom Specialties:1221
Poolside Carts:1320
Formal Dining Specials:1340
Reservation Logs:1410
Plumbing Supplies:1423
Office Equipment:1361
Carpeting Services:1395
Auto Maintenance:1544
Pianos and Violins:1416
```

The Paste Command

The paste command combines files line by line, whereas the cat command appends data to the end of the file. When you use paste to combine two files into a third file, the first line of the third file contains the first line of the first file followed by the first line of the second file. For example, Becky has the file vegetables in her home directory. Its contents are:

```
Carrots
Spinach
Lettuce
Beans
```

She also has the file bread in her home directory. Its contents are:

```
Whole wheat
White bread
Sourdough
Pumpernickel
```

After she executes the command paste Vegetables Bread > Food, the vegetables and bread files are combined, line by line, into the file food. The food file's contents are:

```
Carrots    Whole wheat
Spinach    White bread
Lettuce    Sourdough
Beans      Pumpernickel
```

Note: The paste command normally sends its output to the screen. To capture it in a file, use the redirection symbol.

As you can see, the paste command is most useful when you combine files that contain columns of information. When paste combines items into a single line, it separates them with a tab. For example, look at the first line of the food file:

```
Carrots    Whole wheat
```

When paste combined "Carrots" and "Whole wheat," it inserted a tab between them. You can use the –d option to specify another character as a delimiter. For example, to insert a comma between the output fields instead of a tab, Becky types the paste command:

```
paste –d',' vegetables bread > food
```

After Becky's command executes, the food file's contents are:

```
Carrots,Whole wheat
Spinach,White bread
Lettuce,Sourdough
Beans,Pumpernickel
```

Now you can use the paste command to combine the two product files in one. (Use paste instead of cat, because you're combining fields from two or more files.)

To use the paste command to combine files:

1 After the $ command prompt, type **paste product1 product2** and press **Enter**.

This means "combine the file called product1 with the file called product2." You see the list of product descriptions:

```
Lobby Furniture:1201     Plumbing Supplies:1341
Ballroom Specialties:1221     Office Equipment:1361
Poolside Carts:1320     Carpeting Services:1395
Formal Dining Specials:1340     Auto Maintenance:1544
Reservation Logs:1410     Pianos and Violins:1416
```

Using the Cut Command to Remove Fields

You have learned that files can consist of records, fields, and characters. You may want to retrieve some, but not all, fields in a file. You can use the cut command to remove specific columns or fields from a file. The syntax of the cut command is:

Syntax **cut** -f *list* [-d *char*] *file1 file2 …*
Or
cut -c list *file1 file2 …*

Dissection

- **-f** specifies that you are referring to fields.
- ***list*** is a comma- or hyphen-separated list of integers or range of integers that specifies the field. For example, -f 1 indicates field 1, –f 1,14 indicates fields 1 and 14, and –f 1-14 indicates fields 1 through 14.
- **-d** indicates that a specific character separates the fields.
- ***char*** is the character used as the field separator (delimiter), for example, a comma. The default field delimiter is the Tab character.
- ***file1, file2*** are the files from which you want to cut columns or fields.
- **-c** references character positions. For example, -c 1 specifies the first character and –c 1,14 specifies characters 1 and 14.

Recall the vegetables and bread files in Becky's home directory. She also has the file meats. When she uses the command paste Vegetables Bread Meats > Food, the contents of the food file are:

```
Carrots    Whole wheat    Turkey
Spinach    White bread    Chicken
Lettuce    Sourdough      Beef
Beans      Pumpernickel   Ham
```

Becky wants to extract the second column of information (the bread list) from the file and display it on the screen. She types the following command:

```
cut –f2 food
```

The option –f2 tells the cut command to extract the second field from each line. Tab delimiters separate the fields, so cut knows where to find the fields. She sees the following output on the screen:

```
Whole wheat
White bread
Sourdough
Pumpernickel
```

She extracts the first and third columns from the file with the command:

```
cut –f1,3 food
```

The results of the command are:

```
Carrots    Turkey
Spinach    Chicken
Lettuce    Beef
Beans      Ham
```

Now you can complete your work with the Dominion phone number files by extracting a list of names from the files. First, you will create two files: corp_phones1 and corp_phones2. The corp_phones1 file includes five records of variable size, and a colon separates each field in the record. (Figure 4-1(c) illustrates this type of file structure.) The corp_phones2 file also includes five records of fixed length (the type of file structure illustrated in Figure 4-1(b)). Figure 4-3 illustrates the contents of the two files. You can use the cut command with either file to extract a list of names.

```
File Name:  Corp_phones1 (Variable Size Records - Fields separated by colon :)

219:432:4567:Harrison:Joel:M:4540:Accountant:09-12-1985
219:432:4587:Mitchell:Barbara:C:4541:Admin Asst:12-14-1995
219:432:4589:Olson:Timothy:H:4544:Supervisor:06-30-1983
219:432:4591:Moore:Sarah:H:4500:Dept Manager:08-01-1978
219:432:4527:Polk:John:S:4520:Accountant:09-22-1998

Storage space = 279 bytes
```

```
File Name:  Corp_phones2 (Fixed length records)

Character positions
1-3 5-7 9-12 14-25       26-35    36 38-41 43-50          59-68
====================================================================
219 432 4567 Harrison    Joel      M 4540 Accountant      09-12-1985
219 432 4587 Mitchell    Barbara   C 4541 Admin Asst      1-14-19952
219 432 4589 Olson       Timothy   H 4544 Supervisor      06-30-1983
219 432 4591 Moore       Sarah     H 4500 Dept Manager    08-01-1978
219 432 4527 Polk        John      S 4520 Accountant      09-22-1998

Storage space = 345 bytes
```

Figure 4-3: Two versions of the company telephone file

To create the corp_phones1 and corp_phones 2 files:

1 Use the vi or Emacs editor to create the file corp_phones1.

2 Type the following lines of text, exactly as they appear. Press **Enter** at the end of each line:

```
219:432:4567:Harrison:Joel:M:4540:Accountant:09-12-1985

219:432:4587:Mitchell:Barbara:C:4541:Admin Asst:12-14-1995

219:432:4589:Olson:Timothy:H:4544:Supervisor:06-30-1983

219:432:4591:Moore:Sarah:H:4500:Dept Manager:08-01-1978

219:432:4527:Polk:John:S:4520:Accountant:09-22-1998
```

3 Save the file and create a new file named corp_phones2.

4 Type the following lines of text, exactly as they appear. Consult Figure 4-3 for the precise position of each character. Press Enter at the end of each line.

```
219 432 4567 Harrison    Joel      M 4540 Accountant      09-12-1985
```

```
219 432 4587 Mitchell    Barbara   C 4541 Admin Asst      12-14-1995

219 432 4589 Olson       Timothy   H 4544 Supervisor      06-30-1983

219 432 4591 Moore       Sarah     H 4500 Dept Manager    08-01-1978

219 432 4527 Polk        John      S 4520 Accountant      09-22-1998
```

5 Save the file and exit the editor.

You want to extract the first and last names from the corp_phones1 file first. This file includes variable-length records and fields separated by colon characters. You can select the fields you want to cut by specifying their positions and separator character.

To use the cut command to extract fields from variable-length records:

1 After the $ command prompt, type **cut –f4-6 –d: Corp_phones1** and press **Enter**.

This command means "cut the fields (-f) in positions four through six (4–6) that the colon character (-d:) delimits in the corp_phones1 file."

You see the list of names:

```
Harrison:Joel:M
Mitchell:Barbara:C
Olson:Timothy:H
Moore:Sarah:H
Polk:John:S
```

Now you can extract the first and last names from the corp_phones2 file. This file includes fixed-length records, so you can cut by specifying character positions.

To use the cut command to extract fields from fixed-length records:

1 After the $ command prompt, type **cut –c14-25,26-35,36 corp_phones2** and press **Enter**.

This command means "cut the characters (-c) in positions 14 through 25, 26 through 35, and position 36 (14-25,26-35,36) in the corp_phones2 file."

You see the list of names:

```
Harrison      Joel      M
Mitchell      Barbara   C
Olson         Timothy   H
Moore         Sarah     H
Polk          John      S
```

tip

Make sure not to include a space in the code sequence after the dash (-) options in the cut command. For example, the correct syntax is cut (space) –c14-25,26-35,36 (space)/Corp_ /phones2.

Using the cut command with variable-length or fixed-length records produces similar results. Cutting from fixed-length records creates a more legible display, but requires more storage space. For example, corp_phones2 requires 345 bytes and corp_phones1 requires 279.

Using the Sort Command

Use the sort command to sort a file's contents alphabetically or numerically. UNIX displays the sorted file on the screen by default, but you can specify that you want to store the sorted data in a particular file.

The sort command offers many options, which Appendix B, "Syntax Guide to UNIX Commands," completely describes. Here is an example of its use:

```
sort file1 > file2
```

In this example, the contents of file1 are sorted and the results stored in file2. (If the output is not redirected, sort displays its results on the screen.) Here is a more complex example:

```
sort +.10 file1 > file2
```

This command specifies a sorting key. A sorting key is a field or character position within each line. The sort command sorts the lines based on the sorting key. The + indicates that sorting does not begin at the first character position, but is offset elsewhere in the file. The period (.) indicates that the offset is measured in characters. (If the period is missing, the offset is measured in fields.) The number of characters to offset follows the period. The sample command tells the sort command that the sort key begins at character position 10.

Sorting the corp_phones1 file is relatively easy, because you can refer to field numbers. As you will see when you perform the next steps, sorting the fixed-length file, corp_phones2, is more difficult. In the first two steps, you sort the corp_phones2 file by last name and first name, respectively. In the third and fourth steps, you do the same thing with corp_phones1. Notice that the output of these four steps goes to stdout (the screen). The last step uses the –o option, instead of output redirection, to write the sorted output to a new disk file, sorted_phones.

To sort the phones2 file:

1 After the $ command prompt, type **sort +.13 corp_phones2** and press **Enter**.

This sorts the file by last name, starting at character position 13 (+.13). You see the following on your screen:

```
219 432 4567 Harrison    Joel     M 4540 Accountant      09-12-1985
219 432 4587 Mitchell    Barbara  C 4541 Admin Asst      12-14-1995
```

```
219 432 4591 Moore       Sarah     H 4500 Dept Manager    08-01-1978
219 432 4589 Olson       Timothy   H 4544 Supervisor      06-30-1983
219 432 4527 Polk        John      S 4520 Accountant      09-22-1998
```

2 Type **sort +.25 corp_phones2** and press **Enter**.

This sorts the file by first name, starting at character position 25 (+.25). You see the following on your screen:

```
219 432 4587 Mitchell    Barbara   C 4541 Admin Asst      12-14-1995
219 432 4567 Harrison    Joel      M 4540 Accountant      09-12-1985
219 432 4527 Polk        John      S 4520 Accountant      09-22-1998
219 432 4591 Moore       Sarah     H 4500 Dept Manager    08-01-1978
219 432 4589 Olson       Timothy   H 4544 Supervisor      06-30-1983
```

3 Type **sort –t:+3 corp_ phones1** and press **Enter**.

This sorts the variable-length records (-t:) starting at the Last Name field (+3). You see the following on your screen:

```
219:432:4567:Harrison:Joel:M:4540:Accountant:09-12-1985
219:432:4587:Mitchell:Barbara:C:4541:Admin Asst:12-14-1995
219:432:4591:Moore:Sarah:H:4500:Dept Manager:08-01-1978
219:432:4589:Olson:Timothy:H:4544:Supervisor:06-30-1983
219:432:4567:Polk:John:S:4520:Accountant:09-22-1998
```

4 Type **sort –+: +4 corp_phones1** and press **Enter**.

This sorts the variable-length records (-+: indicates the fields are delimited by a colon) starting at the First Name field (+4). You see the following on your screen:

```
219:432:4587:Mitchell:Barbara:C:4541:Admin Asst:12-14-1995
219:432:4567:Harrison:Joel:M:4540:Accountant:09-12-1985
219:432:4567:Polk:John:S:4520:Accountant:09-22-1998
219:432:4591:Moore:Sarah:H:4500:Dept Manager:08-01-1978
219:432:4589:Olson:Timothy:H:4544:Supervisor:06-30-1983
```

5 To sort by first name and create the output file Sorted_phones, type **sort –t: +4 –0 sorted_phones corp_phones1** and press **Enter**. This sorts the phones1 file by first name and creates an output file, sorted_phones.

Putting It All Together

Now you can use the many file processing tools you've learned all at once. First, use the cat command to create the vendors file. The records in the vendor names file consist of two colon-separated fields: the vendor number and vendor name.

To create the vendors file:

1 After the $ command prompt, type **cat>vendors** and press **Enter**.

2 Type the following text, pressing **Enter** at the end of each line:

```
1201:Cromwell Interiors
1221:Design Extras Inc.
1320:Piedmont Plastics Inc.
1340:Morgan Catering Service Ltd.
1350:Pullman Elevators
1360:Johnson Office Products
```

3 Press **Ctrl+Z**.

Figure 4-4 shows the two files that Worldwide can use to determine which product each vendor supplies.

```
File Name: vendors

Vendor  Vendor Name
Number
=================================
1201:Cromwell Interiors
1221:Design Extras Inc.
1320:Piedmont Plastics Inc.
1340:Morgan Catering Service Ltd.
1350:Pullman Elevators
1360:Johnson Office Products
```

```
File Name: products

Prod     Product     Vendor
Number  Description  Number
=================================
S0107:Lobby Furniture:1201
S0109:Ballroom Specialties:1221
S0110:Poolside Carts:1320
S0130:Formal Dining Specials:1340
S0201:Reservation Logs:1410
```

Figure 4-4: Vendors and products

In the next steps, you use the cut, paste, and sort commands to create a single vendor report for Worldwide Hotels. Start by using the cut command to extract product descriptions and vendor numbers from the products file and storing them in separate files, p1 and p2. Then extract vendor numbers and names from the vendors file, and store them in v1 and v2. Use the paste command to combine the two vendor files (v1 and v2) in a third file, v3. Then combine the two product files (p1 and p2) in a file, p3. Sort and merge the v3 and p3 files, and send their output to the vrep file, the vendor report.

Next use the cat command to create the products file. The records in the products file consist of three colon-separated fields, the product number, the product description, and the vendor number.

To create the products file:

1 After the $ command prompt, type **cat > products** and press **Enter**.

2 Type the following text, pressing **Enter** at the end of each line:

```
S0107:Lobby Furniture:1201
S0109:Ballroom Specialties:1221
S0110:Poolside Carts:1320
S0130:Formal Dining Specials:1340
S0201:Reservation Logs:1410
```

3 Type **Ctrl+Z** to end the cat command

To use the cut, paste, and sort commands to create a report:

1 After the $ command prompt, type **cut –f2 –d: products > p1** and press **Enter**.

This means "extract the data from the second field delimited by a colon in the products file, and store it in the p1 file." It stores these product descriptions in the p1 file:

```
Lobby Furniture
Ballroom Specialties
Poolside Carts
Formal Dining Specials
Reservation Logs
```

2 Type **cut –f3 –d: products > p2** and press **Enter**.

This means "extract the data from the third field delimited by a colon in the products file, and store it in the p2 file." It stores these vendor numbers in the p2 file:

```
1201
1221
1320
1340
1410
```

3 Type **cut –f1 –d: vendors > v1** and press **Enter**.

This means "extract the data from the first field delimited by a colon in the vendors file, and store it in the v1 file." It stores these vendor numbers in the v1 file:

```
1201
1221
1320
1340
1350
1360
```

4 Type **cut –f2 –d: vendors > v2** and press **Enter**.

This means "extract the data from the second field delimited by a colon in the vendors file, and store it in the v2 file." It stores these product descriptions in the v2 file:

```
Cromwell Interiors
Design Extras Inc.
Piedmont Plastics Inc.
Morgan Catering Service Ltd.
Pullman Elevators
Johnson Office Products
```

5 Type **paste v1 v2 > v3** and press **Enter**.

This means "combine the data in v1 and v2, and direct it to the file v3. It stores these vendor numbers and product descriptions in the v3 file:

```
1201    Cromwell Interiors
1221    Design Extras Inc.
1320    Piedmont Plastics Inc.
1340    Morgan Catering Service Ltd.
1350    Pullman Elevators
1360    Johnson Office Products
```

6 Type **paste p2 p1 > p3** and press **Enter**.

This means "combine the data in p2 and p1, and direct it to a file called p3." It stores these vendor numbers and product descriptions in the p3 file:

```
1201    Lobby Furniture
1221    Ballroom Specialties
1320    Poolside Carts
1340    Formal Dining Specials
1410    Reservation Logs
```

7 Type **sort -o vrep -m v3 p3** and press **Enter**.

This means "sort and merge the data in v3 and p3, and direct the output to a file called vrep." It stores these vendor numbers and product descriptions in the vrep file:

```
1201    Cromwell Interiors
1201    Lobby Furniture
1221    Ballroom Specialties
1221    Design Extras Inc.
1320    Piedmont Plastics Inc.
1320    Poolside Carts
1340    Formal Dining Specials
1340    Morgan Catering Service Ltd.
1350    Pullman Elevators
1360    Johnson Office Products
1410    Reservation Logs
```

You used the cut, paste, and sort commands to extract information from files, combine the information, and then sort and merge the information in a new file.

SUMMARY

- Operating systems support regular files, directories, character special files, and block special files. Regular files contain user information. Directories are system files for maintaining the file system's structure. Character special files are related to serial input/output devices, such as printers. Block special files are related to devices, such as disks.
- Files can be structured in several ways. UNIX stores data, such as letters, product records, or vendor reports, in flat ASCII files. UNIX structures files depending on the kind of data they store and recognizes three kinds of regular files: unstructured ASCII characters, records, and trees.
- When performing commands, UNIX processes data—it receives input from the standard input device and then sends output to the standard output device. UNIX refers to the standard devices for input and output as stdin and stdout, respectively. By default, stdin is the keyboard and stdout is the monitor. Another standard device, stderr, refers to the error file that defaults to the monitor.
- Output from a command may be redirected from stdout to a disk file. Input to a command may be redirected from stdin to a disk file. The error output of a command may be redirected from stderr to a disk file.
- The touch command updates a file's time and date stamps and creates empty files.
- When you manipulate files, you work with the files themselves as well as their content. You can use file manipulation commands to create, delete, copy, move, find, and display files.
- The rmdir command removes a directory.
- The cut command removes specific columns or fields from a file. Select the fields you want to cut by specifying their positions and separator character, or you can cut by character positions, depending on the data's organization.
- To combine two or more files, use the paste command. Where cat appends data to the end of the file, the paste command combines files line by line. You can also use paste to combine fields from two or more files.
- Use the sort command to sort a file's contents alphabetically or numerically. UNIX displays the sorted file on the screen by default, but you can also specify that you want to store the sorted data in a particular file.

REVIEW QUESTIONS

1. The UNIX file deletion command is ________________.
   a. del
   b. remove
   c. rm
   d. kill

2. A regular file contains ________________.
   a. a user program
   b. user information

   c. binary digits
   d. UNIX commands

3. Directories are ________________.
   a. any files with a tree structure
   b. standard output devices
   c. special files for identifying disks
   d. system files for maintaining the file system's structure
4. The stderr device refers to ________________.
   a. keyboard errors
   b. the standard error output device
   c. the line printer
   d. an output file
5. You can use the redirection operators to ________________.
   a. change the file type
   b. redirect stdin, stdout, and stderr to a disk file
   c. combine two or more files
   d. merge sorted data
6. The < operator redirects ________________.
   a. standard output
   b. standard input
   c. standard error
   d. none of the above
7. The 2> operator redirects ________________.
   a. standard output
   b. standard input
   c. standard error
   d. none of the above
8. The UNIX move command ________________.
   a. is identical to the copy command
   b. moves only one file at a time
   c. moves multiple files at once
   d. serves the same purpose as a rename command
9. The touch command ________________.
   a. updates a file's access date and time
   b. updates a file's modification date and time
   c. updates a file's creation date and time
   d. both a and b
10. The command to recursively remove directories and their contents is ________________.
    a. rmdir
    b. rmdir –r
    c. rm –r
    d. recurs
11. Use the cut command to ________________.
    a. extract characters or fields from a file
    b. delete characters or fields from a file
    c. truncate records in a file
    d. reverse the paste command's effects

12. Use the paste command to ________________.
   a. place information on the clipboard
   b. combine several files
   c. reverse the cut command's effects
   d. work with the cat command

13. The plus symbol (+) is an option used with the sort command to ________________.
   a. offset the sort field
   b. indicate that the starting field should be skipped
   c. add more options to the sort command
   d. a and b

14. UNIX file processing treats files as ________________.
   a. a sequence of fixed-length records
   b. a sequence of bytes
   c. a sequence of variable-length records
   d. key fields to index records

EXERCISES

1. Explain the difference between cat and paste.
2. What does the command rm –r do that rmdir does not?
3. Describe two ways to create an empty file.
4. Write the command line that sorts the records in the corp_phones1 file by date of hire, the last field in the record. Refer to Figure 4-3.
5. Write the command line for the sort you did in Exercise 4, but use the corp_phones2 file instead of the corp_phones1 file.
6. Suppose you have three files: sales1, sales2, and sales3. Write the paste command that combines these files and separates the fields on each line with a ! character. The command should store the results in the file sales4.
7. Write the cut command that extracts the second field from each line of the sales4 file (that you created in Exercise 6). The command should store its results in the file sales5.
8. Write a command that Becky uses to search for the file johnson_account. She knows the file is somewhere in a directory under her home directory.

DISCOVERY EXERCISES

1. Use the cut command to create the file prod_desc using the products file shown in Figure 4-4. The only field in the file should be the product description.
2. Create a file similar to the products file (Figure 4-4) named prod_desc1. The prod_desc1 file should have the same fields as the products file but contain different data. Use the sort command with the merge option to create an output file, merged_product. Use the input files, prod_desc1 and products.
3. Use any combination of cut and paste to create a new file using the input file, corp_phones2, shown in Figure 4-3. Place the employee's last name, first name, and middle initial in fields 1, 2, and 3.

L E S S O N B

o b j e c t i v e s

After studying this lesson, you should be able to:

- **Create a script file**
- **Use the join command to link files using a common field**
- **Use the awk command to create a professional-looking report**

Assembling Extracted Information

Using Script Files

As you have seen, command-line entries can become long, depending on the number of options you need to use. You can use the shell's command-line history retrieval feature to recall and re-execute past commands. This feature works well for you, but others who need to execute your commands cannot access them repeatedly. MS-DOS users resolve this problem by creating batch files. UNIX users do the same: they create **shell script files** to contain command-line entries. Like MS-DOS batch files, script files contain commands that can be run sequentially as a set. A good candidate for a script file is the series of cut, paste, and sort commands that you entered in Lesson A. You can use the vi editor to create the script file and then make the script executable with the chmod and x commands.

Rolfe Williams is delighted with your vendor report but wants a way to generate the report whenever the data changes. You can create a script that includes a series of commands for creating the file ven_report.

To use the vi editor to create a script:

1 Change your working directory to the corp_dbdirectory, under your home directory.

2 After the $ command prompt, type **vi ven_report** and press **Enter**.

The vi editor starts and creates a new file, ven_report.

3 Enter insert mode and then type the following, pressing **Enter** at the end of every line:

```
a
cut –f2 -d: products > p1
cut –f3 -d: products > p2
cut –f1 -d: vendors  > v1
cut –f2 -d: vendors  > v2
paste v1 v2 > v3
paste p2 p1 > p3
sort –t: -o vrep –m v3 p3
```

These are the same commands you used in Lesson A to create the first vendor report.

4 Press **Esc**.

5 Type **:wq** to exit the vi editor.

Now you can make the script executable with the chmod command. The chmod command sets file permissions. In the example that follows, the chmod command and its ugo+x option make the ven_report file executable by **u**sers (owners), **g**roup, and **o**thers. Run a test to make sure it works.

To make the script executable:

1 After the $ command prompt, type **chmod ugo+x ven_report** and press **Enter**.

See "Setting File Permissions" in Chapter 2 for more information on the chmod command.

2 To make sure the script works, type **./ven_report** and press **Enter**.

You see the same list of vendor numbers and product descriptions that you saw in Lesson A.

Using the Join Command

The join command differs from the other file processing commands in this chapter, because it is used in relational database processing. **Relational databases** consider files as tables and records as rows. They also refer to fields as columns that can be joined to create new records. The join command is the UNIX method that lets you extract information from two files sharing a common field.

For example, you can use the join command to combine information in the vendor and product files, thereby producing the vendor report for Worldwide's purchasing department. Both the vendor and product reports include the vendor number. The syntax of the join command is:

Syntax **join** [options] *file1 file2*

Dissection

- ***file1, file2*** are two input files that must be sorted on the join field—the field you want to use join the files. The join field is also called a **key**. You must sort the files before you can join them. When you issue the join command, UNIX compares the two fields. Each output line contains the common field followed by a line from *file1*, followed by a line from *file2*. You can modify output using the options described next. If records with duplicate keys are in the same file, UNIX joins on all of them. You can create output records for unpairable lines, for example, to append data from one file to another without losing records.

Options

- **-j** specifies the common fields on which the join is to be made.
- **-o** specifies a list of fields to be output. The list contains blank-separated field specifiers in the form *m.n*, where *m* is the file number and *n* is the position of the field in the file. Thus –o 1.2 means "output the second field in the first file."
- **-t** specifies the field separator character. By default this is a blank, tab, or new line character. Multiple blanks and tabs count as one field separator.
- **-a *n*** produces a line for each unpairable line in file *n*, where *n* = 1 or 2.
- **-e *str*** replaces the empty fields for the unpairable line in the string specified by *str*. The string is usually a code or message to indicate the condition; that is, –e "No Vendor Record."

For an example of join logic, see Figure 4-5.

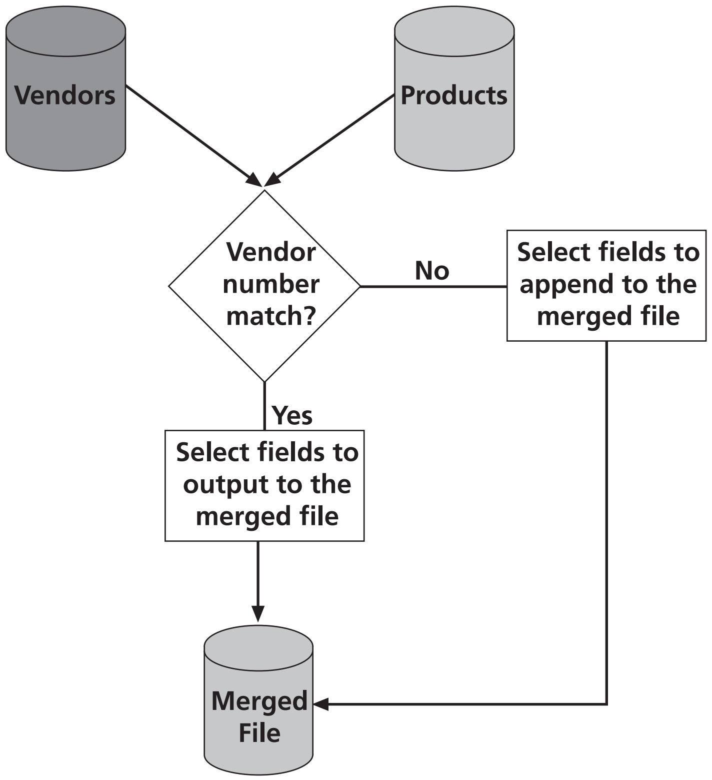
Vendors
Products
Vendor number match?
No
Select fields to append to the merged file
Yes
Select fields to output to the merged file
Merged File

Figure 4-5: Relational join

In Figure 4-5, the join command combines fields from the vendor and product files. Then it searches for vendor numbers in the vendor file matching those in the product file. If a field does not match, UNIX selects it and later appends it to the merged file. If the field does match, UNIX sends it to the merged file.

Using the Join Command to Create the Vendor Report

You can use the join command to create a vendor report showing what products Worldwide's purchasing department has in stock.

To use the join command to create a report:

1. After the $ command prompt, type **join –a1 –e "No Products" –j1 1 –j2 3 –o 1.2 2.2 –t: vendors products > vreport**; **cat vreport** and press **Enter**.

   In this command, the –j option indicates the first or second specified file, such as vendors or products. Numbers following j1 and j2 specify field numbers used for the join or match. Here, UNIX uses the first field of the vendors record to join the third field of the products file.

The –a option tells the command to print a line for each unpairable line in the file number. In this case, a line prints for each vendor record that does not match a product record.

The –e option lets you display a message for the unmatched (-a 1) record, such as “No Products.”

The –o option sets the fields that will be output when a match is made.

The 1.2 indicates that field two of the vendors file is to be output along with 2.2, field two of the products file.

The –t option specifies the field separator, the colon. This join command redirects its output to a new file, vreport. The cat command displays the output on the screen.

The report contains this information:

```
Cromwell Interiors:Lobby Furniture
Design Extras Inc.:Ballroom Specialties
Piedmont Plastics Inc.:Poolside Carts
Morgan Catering Service Ltd.:Formal Dining Specials
Pullman Elevators:No Products
Johnson Office Products:No Products
```

A Brief Introduction to Awk

Awk, a pattern-scanning and processing language, helps to produce professional-looking reports. Although you can use the cat and more commands to display the output file that you create with your join program, the awk command (which starts the Awk program when you type it on the command line) lets you do the same thing more quickly and easily. The syntax of the awk command is:

Syntax **awk** [-Fsep] ‘ *pattern* {*action*} ..’ *filenames*

Dissection

- **awk** checks to see if the input records in the specified files satisfy the ***pattern*** and, if they do, awk executes the ***action*** associated with it. If no ***pattern*** is specified, the ***action*** affects every input record.
- **-F:** means the field separator is a colon.

To generate and format the vendor report:

1 After the $ command prompt, type **awk –F: ‘{printf “%-28s\t %s\n”, $1, $2}’ vreport** and then press **Enter**. This command is explained in detail after Step 2.

2 You see the vendor report, including vendor names and product descriptions:

```
Cromwell Interiors             Lobby Furniture
Design Extras Inc.             Ballroom Specialties
Piedmont Plastics Inc.         Poolside Carts
Morgan Catering Service Ltd.   Formal Dining Specials
Pullman Elevators              No Products
Johnson Office Products        No Products
```

The parts of the awk command you typed in Step 1 are:

- **awk –F:** calls the Awk program and identifies the field separator as a colon.
- **'{printf "%-28s\t %s\n", $1, $2}'** represents the *action* to take on each line that is read in. Single quotes enclosed the action.
- **printf** is a print formatting function from the C language. It lets you specify an edit *pattern* for the output. The code inside the double quotes defines this pattern. The code immediately following the % tells how to align the field to be printed. The – sign specifies left alignment. The number that follows, 28, indicates how many characters you want to display. The trailing s means that the field consists of non-numeric characters, also called a "string." The \t inserts a tab character into the edit pattern. The %s specifies that another string field should be printed. You do not need to specify the string length in this case, because it will be the last field printed (the product name). The \n specifies to skip a line after printing each output record. The $1 and $2 separated with a comma indicates that the first and second fields in the input file should be placed in the edit pattern where the two s's appear. The first field is the vendor name, and the second is the product description.
- **vreport** is the name of the input file.

Using the Awk Command to Refine the Vendor Report

To refine and automate the vendor report, you can create a shell script, as you did at the beginning of this lesson. This new script, however, includes only the awk command, not a series of separate commands. You then call the Awk program using awk with the –f option. This option tells awk that the code is coming from a disk file, not from the keyboard. You can present the *action* statements inside the Awk program file in a different way. The program file includes additional lines needed to print a heading and the current date for the report.

The next steps show what happens when you enter the Awk program in a file like this. You use the FS variable to tell the program what the field separator is, in this example, a colon. FS is one of many variables that awk uses to advise the program about the file being processed. Other codes you see here set up an initial activity that executes once when the program loads. BEGIN followed by the opening curly brace ({) indicates this opening activity. The closing curly brace (}) marks the end of actions performed when the program first loads. These *actions* print the headings, date, and dash lines that separate the heading from the body of the report.

To create the awk script:

1 After the $ command prompt, type **vi awrp** and press **Enter** to start the vi editor and create the file awrp.

2 Type the code:

```
BEGIN {
        { FS = ":"}
        { print "\t\tVendors and Products" }
        { "date" | getline d }
        { printf "\t   %s\n",d }
        { print "Vendor Name\t\t\t Product Names" }
        {print"===========================================\n" }
       }
      { printf "%-28s\t%s\n",$1, $2 }
```

3 Press **Esc**.

4 Type **:wq** to exit the vi editor.

5 After the $ command prompt, type **awk –f awrp vreport > v_report**, and press **Enter**.

This means "using the Awk program, combine the fields from the awrp file with the fields from the vreport file, and send them to a new file called v_report."

6 Type **cat v_report** and press **Enter**.

You see the following report:

```
       Vendors and Products
    Sun Dec 20 21:03:41 EST 1998
Vendor Name                    Product Names
=======================================================

Cromwell Interiors             Lobby Furniture
Design Extras Inc.             Ballroom Specialties
Piedmont Plastics Inc.         Poolside Carts
Morgan Catering Service Ltd.   Formal Dining Specials
Pullman Elevators              No Products
Johnson Office Products        No Products
```

7 To print the report on the default printer, type **lpr v_report** and press **Enter**.

You produced a vendor and product name report and then formatted and printed the report.

SUMMARY

- To automate command processing, include commands in a script file that you can later execute like a program. Use the vi editor to create the script file and the chmod command to make it executable.
- Use the join command to extract information from two files sharing a common field. You can use this common field to join the two files. You must sort the two files on the join field—the one you want to use to join the files. The join field is also called a "key." You must sort the files before you can join them.
- Awk is a pattern-scanning and processing language useful for creating a formatted and professional-looking report. You can enter the Awk language instructions in a program file using the vi editor and call it using the awk command.

COMMAND SUMMARY

Chapter 4 commands

Command	Purpose	Options covered in this chapter
awk	Start the Awk program to format output	-F identifies the field separator –f indicates code is coming from a disk file, not the keyboard
cp	Copy one or more files	
cut	Extract specified columns or fields from a file	-c refers to character positions -d indicates a specified character separates the fields -f refers to fields
find	Find files	-name specifies the name of files you want to locate
join	Combine files having a common field	-a *n* produces a line for each unpairable line in file *n* -e *str* replaces the empty fields for an unpairable file with the specified string -j uses specified common fields when joining -o outputs a specified list of fields -t indicates a specified character separates the fields

Chapter 4 commands (*continued*)

less	Display the file's contents, pausing at the end of each screen	
lpr	Print the command output on the default printer	
more	Display the contents of a file, pausing at the end of each screen	
mv	Move one or more files	
paste	Combine fields from two or more files	
rm	Remove one or more files	–i specifies that UNIX should request confirmation of file deletion before removing the files –r specifies that directories should be recursively removed
rmdir	Remove a directory	
sort	Sort the file's contents	+*n* sorts on the field specified by *n* .+ designates the position that follows an offset (+) as a character position, not a field position –t indicates that a specified character separates the fields –m means to merge files before sorting –o redirects output to the specified file
touch	Update an existing file's date and time stamp command or creates empty new files	–a specifies that only the access date and time is to be updated –m specifies that only the modification date and time is to be updated –c specifies that no files are to be created

REVIEW QUESTIONS

1. Which of these statements is false?
   a. The join program requires that you sort files by the join field.
   b. A script file is similar to an MS-DOS batch file.
   c. The Awk program eliminates the need to write high-level languages.
   d. You can make your script files executable.
2. The join command ________________.
   a. combines files with a common field
   b. can only be used with merged files
   c. is the opposite of the paste command
   d. formats output
3. Used with the awk command, the –F option specifies ________________.
   a. an external file containing instructions
   b. that the input file has fixed-length records
   c. the output's form
   d. none of the above
4. The BEGIN script in the Awk program ________________.
   a. executes only once during the program cycle
   b. is normally enclosed in curly braces if you issue more than one command
   c. executes when the program loads
   d. all of the above
5. Which option codes does the join command use to specify the list of output fields?
   a. –o
   b. –f
   c. –x
   d. –e
6. Which option does the join command use to specify the string to print when a line is unmatched?
   a. –e
   b. –o
   c. –f
   d. –x
7. Used with the join program, which of these displays the second field in the first file?
   a. –o 2.1
   b. –o 1.1
   c. –o 1.2
   d. none of the above
8. The join command is like a database program in that both ________________.
   a. must work with a relational database file
   b. access tables instead of files
   c. use a relational join operation to connect files
   d. must be read by database-only programs

EXERCISES

1. What is the advantage of using script files?
2. What is the difference between join and paste?
3. Why is the operation performed by the join command called a "relational join?"
4. What two ways do you use to specify patterns and actions to the awk command?
5 Create a join script for the vendors and products files showing which products do not have a matching vendor record.
6. Write a simple Awk program to print the phones1 file in Figure 4-3.
7. Write a script file to display the vendor file, and then run the script.

DISCOVERY EXERCISES

1. Use vi, Emacs, or the method of your choice to create the files cust_names, cust_ids, and cust_status. The files should have the following contents.

   Contents of cust_names:

   ```
   Smith Furniture Co.
   Wells Manufacturing
   Rose Department Store
   Haywood Resort
   ```

   Contents of cust_ids:

   ```
   101
   102
   114
   197
   ```

   Contents of cust_status:

   ```
   ACTIVE
   ACTIVE
   INACTIVE
   ACTIVE
   ```

2. Use the paste command to create a file named cust1. Combine the cust_ids and cust_names files to create cust1. Use the colon character as the field delimiter.
3. Use the paste command to create a file called cust2. Combine the cust_ids and cust_status files to create cust2. Use the colon character as the field delimiter.
4. Use the join command to join cust1 and cust2 and create the file cust_info. The key field is the first field of cust1 and cust2 (which lists the customer ID).
5. Write an awk command that reads the cust_info and displays the customer IDs followed by the customer names. The customer IDs should be printed in a right-justified field of 10 spaces.